

Trabajo de Introducción a la Investigación en Visión Artificial.

Estado del Arte de VisualSLAM

Estudiante: Elías Barcia Mejias

Universidad Rey Juan Carlos
Madrid, España

Profesor:José María Cañas Plaza

Colaborador:Eduardo Perdices García

1 Introducción

Actualmente la investigación y desarrollo en robótica móvil está en pleno auge. Los robots modernos están equipados con múltiples sensores y uno de los más utilizados son las cámaras, ya que permiten al robot captar en imágenes todo el entorno que le rodea. En contra partida, el procesado de imágenes conlleva una carga notable de CPU debido a la enorme cantidad de información que puede aportar cada imagen. Una de las funcionalidades más importantes que se persigue, es que los robots móviles puedan desplazarse por su entorno y navegar desde la posición A a la posición B de forma autónoma. Esta tarea no resulta muy complicada en entornos estructurados, donde el robot conoce de antemano el terreno por el que se mueve o sabe de la existencia de alguna baliza que le dé pistas de su posición. Pero en entornos no estructurados, donde el robot desconoce por completo el terreno, carece de mapas y no existe a priori ningún tipo de marca o baliza que pueda guiar al robot, la navegación resulta mucho más compleja. En exteriores, podríamos guiar al robot mediante GPS, pero la señal GPS no llega con la suficiente potencia a todas partes. Por ejemplo en interiores de edificios o en zonas subterráneas, o mejor aún, imaginemos que enviásemos el robot a explorar la superficie del planeta Marte, donde la señal GPS es inexistente. ¿Cómo se las arreglaría el robot para desplazarse por el terreno de forma autónoma sin perderse? Hoy en día ya existe una familia de técnicas que permite al robot navegar de manera autónoma por zonas desconocidas para él, esta técnica se llama VisualSLAM.

Visual SLAM (*Simultaneous Localization and Mapping*) es una técnica utilizada principalmente con robots móviles y que aporta al robot la capacidad de autolocalizarse y generar mapas del entorno que le rodea en tiempo real. Gracias a ese mapa y principalmente a esa autolocalización se pueden utilizar las técnicas de navegación autónoma, que requieren inevitablemente de una estimación de posición propia fiable. VisualSLAM básicamente se comporta como una caja negra que procesa las imágenes en secuencia captadas por una o varias cámaras. A partir de esas imágenes el robot es capaz de obtener su posición 3D en el mundo que le rodea. De esta forma el robot podrá desplazarse en su entorno de forma

autónoma sin perderse.

El robot, debe contar con una capacidad de cálculo suficiente que le permita ejecutar el software de procesado de imágenes y al mismo tiempo realizar la generación de mapas. Estas tareas requieren ser ejecutadas en tiempo real, unos 30 fotogramas por segundo. Es posible utilizar la técnica de VisualSLAM hoy en día en pequeños dispositivos gracias al aumento de su potencia de computación. Dependiendo del tipo de cámaras con las que esté equipado el robot, tendrá mayor o menor capacidad de ejecutar VisualSLAM. Como mínimo el robot debe tener una cámara RGB, muy común en los drones, aunque también puede tener 2 cámaras estéreo que le ayudarán a representar el entorno en 3D con mayor fiabilidad. Otras cámaras, RGBD como las utilizadas en el proyecto Tango se ayudan de un sensor de profundidad que también capacita al robot para representar en tres dimensiones el mundo que les rodea con mayor robustez y precisión.

2 Aplicaciones en VisualSLAM

Hoy en día VisualSLAM ya tiene muchas aplicaciones y aún más que están por llegar en un futuro próximo, a continuación se expondrán varios ejemplos de aplicaciones, desde teléfonos móviles hasta robots aspiradora.

2.1 Proyecto Tango

El proyecto Tango es un proyecto colaborativo que trata de equipar a los smartphones y Tablets con sistema operativo Android la capacidad de medir la profundidad a la que se encuentra cada píxel de las imágenes capturadas por la cámara. Para ello los dispositivos compatibles con Tango dispondrán de 2 cámaras, una cámara RGB y otra que captura la profundidad, así el smartphone es capaz de construir un mapa en 3D del entorno (Figura 1(a)). Los sensores del smartphone son capaces de tomar más de 250 millones de medidas 3D por segundo y con estos datos pueden construir un modelo 3D de los alrededores del teléfono. Las posibilidades que ofrecerán este tipo de dispositivos serán muy variadas, desde medir

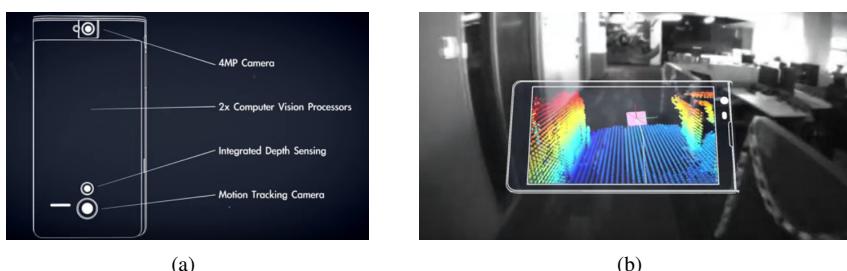


Figure 1: Esquema de prototipo de smartphone Tango (a).Generación de mapa (b)

las dimensiones de la habitación, hasta lo más útil como guiar a personas con discapacidades visuales en el interior de edificios. Pero también tendrá utilidades para el entretenimiento como convertir una habitación en el escenario de un juego mediante realidad aumentada.

Al ser una tecnología nueva aún no hay un elevado número de dispositivos que lo so-

porten. De momento existen 2 móviles compatibles con Tango¹, el Lenovo Phab 2 pro (Figura ??) y el Asus Zenfone AR (Figura ??). En el caso del Zenfone AR estará equipado con 3 cámaras traseras, una para seguir objetos (motion tracking), otra para detectar profundidad y otra de alta resolución de 23 MP. Con estas 3 cámaras el smartphone podrá crear una modelo tridimensional del entorno y seguir su movimiento. La cámara de localización permitirá al ZenFone conocer su posición 3D en todo momento mientras se mueve por el entorno. La cámara de profundidad está equipada con un proyector de Infrarrojos que le permite medir distancias hasta los objetos en el mundo real.

2.2 Magic Plan y Canvas

Magic Plan es una aplicación que permite de forma interactiva obtener planos de habitaciones o del interior de un edificio, utilizando para ello la cámara de nuestra tablet o smartphone, sólo es necesario sacar fotos. Esta aplicación es gratuita, aunque si se desea obtener el plano en formato digital (pdf, jpg, csv y otros) será necesario pagar una pequeña cantidad de dinero. Es muy sencilla de utilizar y en cuestión de minutos se obtiene un plano fiable (Figura 2(a)) sin necesidad de medir, dibujar, mover muebles y sin necesidad de ser un experto. La aplicación utiliza técnicas de VisualSLAM y se apoya también en la información de los giroscopios de los dispositivos. Es compatible con Android y dispositivos Apple. En el caso de Android, actualmente la última versión es compatible con el sistema Tango, por tanto el procedimiento de captura es mucho más sencillo, robusto y preciso ya que permite detectar con mayor exactitud todas las paredes de la habitación, visualizarlas en 3D y aplicar realidad aumentada. Canvas es otro aplicativo diseñado exclusivamente para el Ipad de Apple. Equipado con el sensor Structure, también permite escanear en 3D las habitaciones de la casa.



Figure 2: La pantalla de un smartphone utilizando Magic Plan(a).El sensor Structure en un Ipad(b)

2.3 Pix4D

Pix4D² es un software especializado en fotogrametría. Permite la posibilidad de generar mapas 2D y 3D desde fotografías. Las imágenes pueden ser transmitidas vía wireless a Pix4DDim para procesarlas y convertirlas a mapas 2D y 3D. Posteriormente esta información será accesible desde la nube para poder analizarlas y compartirlas. Pix4D permite crear

¹<https://get.google.com/tango/>

²<https://pix4d.com/>

mapas con exactitud a partir de fotografías de interiores, también tiene aplicaciones en minería para medir superficies y volúmenes (Figura 3(a)) de minas a cielo abierto, incluso se utiliza con finalidades forenses para recrear en 3D escenarios de accidentes, que posteriormente pueden ser analizadas con todo detalle. También tiene aplicaciones en la agricultura para obtener mapas de cosechas utilizando la información que proporcionan las cámaras especiales como la Parrot Sequoia (Figura 3(b)). Con la aplicación Pix4DCapture podremos controlar un dron desde nuestro smartphone para que genere un mapa. El dron puede volar de forma autónoma siguiendo algunas de las trayectoria de vuelo que trae por defecto el producto o también puede generar el mapa mientras lo teledirigimos.



Figure 3: Pix4D cálculo de volumen(a). Flujo de datos de Pix4D(b). Trayectorias que pueden seguir los drones con Pix4DCapture(c). Cámara multiespectral Parrot Sequoia (d)

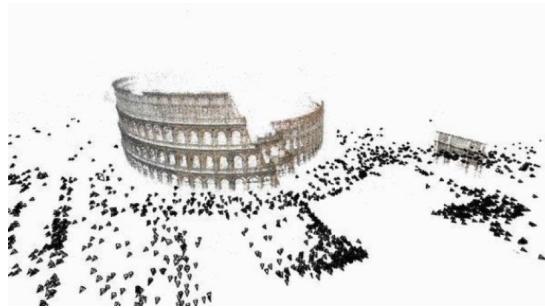
2.4 Photo Tourism

PhotoTourism o Photo Synth es un software inicialmente creado por la universidad de Washington en colaboración con Microsoft. Es un sistema que toma grupos de conjuntos de fotografías disponibles online sobre un lugar en concreto, normalmente sobre un monumento turístico mundialmente conocido (como NotreDame, el Coliseo (Figura 4(a)), La Fontana de Trevi) y es capaz de reconstruir puntos 3D de los monumentos y también calcular o estimar la posición de la cámara desde donde se tomaron las fotografías. Proporciona una nueva forma de navegar a través de fotografías de un destino turístico y una nueva forma de hacer visitas virtuales a monumentos. Este sistema utiliza la técnica de *Structure From Motion* SFM. SFM encuentra coincidencias de puntos característicos entre distintas fotografías de un mismo lugar y que han sido tomadas desde distintos puntos de vista y así es capaz de calcular la localización 3D de dichos puntos característicos y también la localización 3D desde donde se tomaron las fotografías. A diferencia de VisualSLAM, el procesamiento de estas fotografías es offline, sin necesidad de tiempo real, por lo que pueden ser ejecutadas desde un PC que por lo general tiene una capacidad de computación mucho mayor que una tablet o teléfono móvil.

2.5 Aplicaciones en Robótica Móvil

Visual SLAM tiene aplicaciones directas en robótica. Un ejemplo podría ser el Robot Gita de Piaggio.

1. **El robot Gita:** Este novedoso robot (Figura 5) tiene incorporadas varios pares de cámaras estéreo, en la parte trasera y delantera. Con las imágenes captadas por es-



(a)

Figure 4: Recreación del Coliseo de Roma generado con Photo Tourism.

tas cámaras se puede realizar VisualSLAM, además es capaz de seguir a su dueño siempre y cuando el humano lleve un cinturón con otras 2 cámaras estéreo, esta funcionalidad se consigue comparando el SLAM del robot con el SLAM captado por el cinturón. El robot dispone de un compartimento interior o maletero y tiene suficiente potencia como para poder transportar hasta 20 Kg. Podría ser de gran utilidad a la hora de ir al supermercado, ya que el robot nos seguirá transportando la compra en su interior, no necesitaremos el típico carrito, incluso nos permitiría ir al centro comercial en bicicleta. Otra utilidad sería en el interior de un hotel, el robot podría realizar las funciones de camarero y hacer servicio de habitaciones transportando la comida directamente a las habitaciones del hotel. También podría ser un estupendo ayudante para un mecánico, ya que podría transportar la pesadas herramientas o piezas³.



(a)



(b)

Figure 5: El cinturón con cámaras estéreo (a) La capacidad de carga del robot Gita (b)

2. **Reconocimiento de Objetos:** Otra utilidad de SLAM es que mejoran la capacidad de los robots móviles a la hora de reconocer objetos. Los sistemas de reconocimiento de objetos utilizarán la información proporcionada por SLAM para mejorar su capacidad de reconocimiento. La capacidad de reconocimiento será muy útil para aquellos robots que tengan que manipular objetos en su entorno. Con SLAM, los sistemas de reconocimiento pueden tomar como entradas varias imágenes desde distintos puntos de vista, por lo tanto el reconocimiento resulta más sencillo que si tuviesen tan sólo

³<http://spectrum.ieee.org/automaton/robotics/home-robots/piaggio-cargo-robot>

una imagen estática⁴.

3. **Robot Aspirador:** Recientemente ha entrado en los hogares el uso de VisualSLAM gracias a los últimos modelos de aspiradora equipados con cámaras. Estos aspiradores robotizados disponen de cámaras que le permiten obtener un mapa de la habitación o planta del edificio y gracias a este mapa son capaces de aspirar toda la superficie del suelo de la habitación de manera eficiente, sin dejar ninguna zona de la planta sin limpiar. Además están equipados con sensores de proximidad, que les permiten esquivar obstáculos y aunque tengan que modificar su recorrido momentáneamente son capaces de seguir limpiando ya que pueden utilizar el mapa para continuar su ruta. Entre los distintos aspiradores que hay en el mercado podríamos destacar:

- (a) Aspirador Dyson 360 Eye (Figura 6(a))⁵,
- (b) Aspirador Roomba 966 (Figura 6(b))⁶,
- (c) Aspirador LG-Hombot (Figura 6(c))⁷.

Tanto el modelo de Dyson como Roomba utilizan una cámara de 360 grados, en cambio el modelo de LG utiliza una doble cámara, y es capaz de aspirar la casa incluso en la oscuridad.

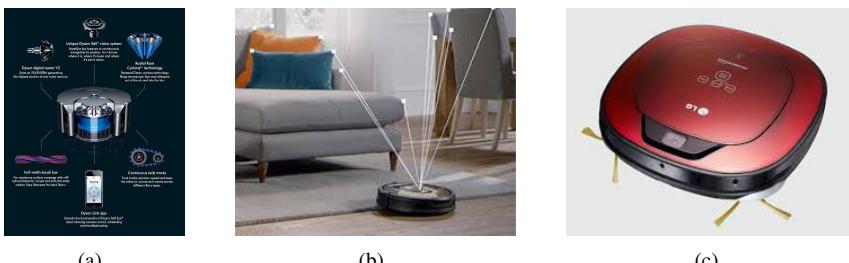


Figure 6: Robot Dyson 360 Eye (a) Robot Roomba 966 (b) Robot Hombot de LG (c).

4. **Drones:** Por último no podemos olvidar los drones, robots voladores equipados con cámara que también pueden obtener mapas de su entorno con VisualSLAM. Existen también proyectos para equipar a drones con dispositivos compatibles con Tango para que puedan obtener mapas de interiores con precisión, robustez y velocidad⁸.

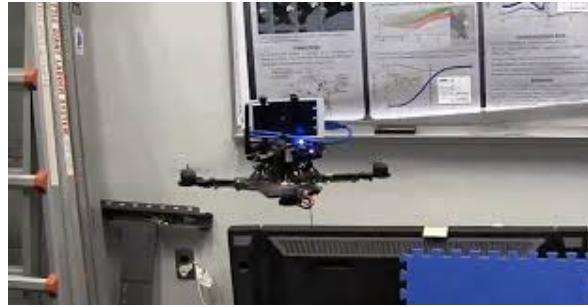
⁴<http://www.roboticsproceedings.org/rss11/p34.pdf>

⁵<http://www.dyson.com>

⁶<http://www.irobot.es/robots-domesticos/aspiracion>

⁷<http://www.lg.com/es/aspiradoras/lg-VR64702LVMT>

⁸<http://spectrum.ieee.org/automaton/robotics/drones/autonomous-quadrotor-flight-based-on-google-project-tango>



(a)

Figure 7: Dron equipado con dispositivo compatible con Tango

3 Problemas de Visual SLAM

Actualmente las técnicas de Visual SLAM presentan algunos problemas o inconvenientes que todavía son difíciles de sortear en la práctica. En esta sección presentaremos algunos de ellos:

3.1 Inicialización del Mapa:

Si en Visual SLAM queremos conseguir una estimación lo más exacta posible de la posición de la cámara es necesario contar con una buena inicialización del Mapa. Se debe contar con un sistema de coordenadas globales definido, y se tomarán puntos de referencia del entorno como puntos iniciales del mapa en el sistema global de coordenadas. Utilizando este método podemos inicializar VisualSLAM en un sistema de coordenadas global en la tierra. La transformación de estos puntos iniciales al sistema de referencia global se realizará mediante homografía. Objetos de referencia como objetos 3D también se han utilizado para obtener un sistema global de coordenadas, posiciones iniciales de la cámara son estimadas gracias al seguimiento de objetos de referencia. En MonoSLAM por ejemplo se utilizan al menos 4 puntos 3D como objetos de referencia, y la forma del objeto se usa para mejorar el mapa.

3.2 Ambigüedad en la escala:

En algunas aplicaciones con Visual SLAM se necesita información de escala absoluta. Para obtener una referencia de escala absoluta se pueden utilizar zonas de la anatomía del usuario, como la cara, su mano o el propio cuerpo. En todos estos métodos se asume que entre personas la diferencia de tamaño es mínima para dichas partes del cuerpo. Se han dado otras aproximaciones como utilizar algunos de los sensores con los que ya están equipados la mayoría smartphones tales como acelerómetros, giroscopios y sensores magnéticos. Para eliminar el ruido de estos sensores se utiliza una técnica de filtro de dominio de frecuencia.

3.3 Dificultad para operar en entornos con pocas texturas:

Visual SLAM utiliza el emparejamiento de píxeles o puntos característicos entre varios frames consecutivos. El emparejamiento suele fijarse en esquinas, bordes o puntos distin-

tivos que fácilmente podrán localizarse entre frames. Pero cuando en el entorno hay pocas texturas o presenta una alta monotonía de texturas, el emparejamiento es difícil de realizar ya que un punto en un fotograma podría corresponder con N puntos en el siguiente fotograma y por tanto se dispararía el error de posición. Quizá este sea uno de los problemas más difíciles de solucionar con VisualSLAM [10].

4 Técnicas de Visual SLAM

En esta sección explicaremos varios de los algoritmos conocidos hasta ahora de Visual SLAM, todos ellos se caracterizan por intentar estimar la posición de la cámara con el menor error posible junto con la generación de un mapeo del entorno que rodea a la cámara. También se explicarán los módulos principales que componen un algoritmo de Visual SLAM. Por último, el conjunto de algoritmos de Visual SLAM se podrían dividir en base al número de regiones que se utilizan en cada frame o imagen recibida para calcular la localización y el mapa. De un lado estarían el grupo de los algoritmos denso / escaso (*dense/sparse*)⁹, por otro lado estarían los Métodos Directos y Métodos Indirectos o basados en características que se caracterizan por el modo en el que se son procesadas las imágenes de entrada (*direct/indirect*).

4.1 Módulos del algoritmo de Visual SLAM

La estructura de un algoritmo genérico de Visual SLAM, podría estar dividida en varios módulos. En esta sección describiremos los 5 componentes que suelen aparecer en la mayoría de algoritmos de Visual SLAM actuales. Estos módulos podrían ser Inicialización, Localización, Generación de Mapa, Optimización global del mapa y Relocalización. Los 3 primeros módulos podrían ser considerados como básicos o esenciales para poder desarrollar correctamente el algoritmo de Visual SLAM, en cuanto a los 2 últimos no son considerados esenciales aunque sí que es muy recomendable que el sistema los tenga incorporados ya que le aportan mayor robustez y precisión al sistema. Los métodos de Visual SLAM utilizan diferentes metodologías para cada módulo, las características de un algoritmo Visual SLAM dependen directamente de la metodología utilizada. Por ello para conocer el rendimiento, ventajas y limitaciones de los algoritmos Visual SLAM es importante entender cada módulo del algoritmo [10].

- 1. Inicialización:** Para iniciar un algoritmo de Visual SLAM es necesario definir un sistema de coordenadas que permita estimar la posición de la cámara y construir el mapa 3D inicial del entorno desconocido, normalmente mediante homografía.
- 2. Localización (Tracking):** Para la localización de la cámara es necesario realizar correspondencias 2D-3D entre la imagen captada y el mapa construido. Esto se consigue utilizando emparejamiento de puntos característicos o seguimiento de puntos en la imagen. La estimación de la posición de la cámara se obtiene mediante el algoritmo iterativo de Gauss-Newton¹⁰. Los parámetros intrínsecos de la cámara suelen estar calibrados de antemano, por tanto son valores conocidos. Para hallar la estimación de la posición de la cámara se aplica una matriz de traslación y rotación sobre los

⁹<https://www.kudan.eu/kudan-news/different-types-visual-slam-systems/>

¹⁰<https://www.youtube.com/watch?v=rpgMZRFm1aI>

parámetros extrínsecos de la cámara y así obtendremos las coordenadas de la cámara en el sistema global de coordenadas.

3. **Generación de Mapa:** El mapa se va generando a medida que se van añadiendo nuevas estructuras 3D de regiones que no habían sido incluidas previamente en el mapa.
4. **Optimización global:** A medida que la cámara se va desplazando se van generando errores acumulativos de estimación de posición. Se puede calcular el error acumulativo desde el inicio hasta la posición actual cuando tras realizar varios movimientos con la cámara volvemos a visitar la imagen inicial o localización de partida. Para ello nos ayudaremos de la técnica de cierre de bucle. El cierre de bucle, es una técnica en la que se trata de encontrar la imagen actual entre las imágenes anteriormente capturadas. Si el bucle es detectado significa que la cámara ha obtenido una imagen que ya había capturado anteriormente. La técnica de cierre de bucle se realiza para obtener una geometría consistente del mapa. La técnica de *Bundle Adjustment* (BA) se utiliza para minimizar el error de retroproyección del mapa mediante optimización de la posición de la cámara y del mapa.
5. **Relocalización:** Este módulo es requerido cuando falla la localización (Tracking) debido a movimientos rápidos o bruscos u occlusiones de la cámara. En este caso se debe encontrar de nuevo la posición de la cámara con respecto al mapa. Si un algoritmo de Visual SLAM no cuenta con este módulo, el sistema dejará de funcionar en caso de que se perdiese por un instante la localización de la cámara. Este módulo también es útil para resolver el problema de secuestro en robótica móvil.

4.2 Métodos Densos y Métodos Escasos

Los Métodos Escasos utilizan sólo un pequeño subconjunto de píxeles de ciertas regiones de la imagen, mientras que los Métodos Densos utilizan la mayoría de los píxeles de la imagen captada. Por tanto los mapas generados por los Métodos Densos proporcionan muchos más detalles de la escena al utilizar muchos más puntos, pero también necesitan de una capacidad de computo muy elevada, de hecho la mayoría de Métodos Densos requieren la utilización de GPU. Los Métodos Escasos, al tratar menos puntos, obtienen unos mapas con muy pocos detalles, más parecido a una nube de puntos, donde principalmente se representan la trayectoria y las diferentes localizaciones que ha ido ocupando la cámara en el mapa 3D.

4.3 Métodos Directos y Métodos Indirectos

Los Métodos Indirectos en Visual SLAM intentan extraer puntos característicos de la imagen y partiendo de estos puntos trata de calcular la posición de la cámara y de generar el mapa. Los puntos característicos pueden ser desde esquinas (obtenidas mediante algoritmo FAST) y bordes hasta otros descriptores de imagen más sofisticados como SIFT o ORB. Sin embargo, los Métodos Directos en Visual SLAM utilizan directamente los valores de intensidad de los píxeles para construir el mapa y calcular la posición de la cámara. En cuanto a los Métodos Indirectos tratan de recuperar la profundidad, estructura del entorno y la posición de la cámara a través de una optimización del mapa y los parámetros de la cámara mediante Bundle Adjustment. En el siguiente gráfico 8(a) se muestran los principales algoritmos de Visual SLAM y que posición ocuparían al clasificarlos entre Métodos Directos e Indirectos

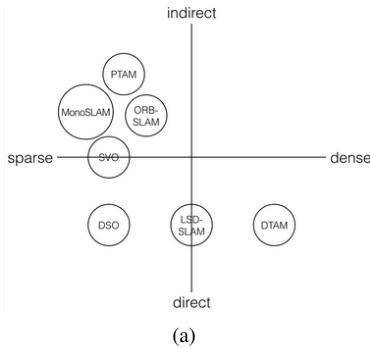


Figure 8: Mapa de clasificación de los principales algoritmos de Visual SLAM.

y Métodos Densos y Escasos. Tres métodos (MonoSLAM, PTAM, ORB-SLAM) podrían clasificarse dentro del grupo Métodos Indirectos y Escasos. El método DSO se podría clasificar como método Escasos y Directos. El método LSD-SLAM podrían clasificarse como un método Directo, a medio camino entre Denso y Escaso. Como Método Denso y Directo estaría el método DTAM. El método SVO podría clasificarse como Método Escaso pero a medias de Método Directo y Método Indirecto. Se puede observar como la zona de Métodos Indirectos y Métodos Densos (cuadrante superior derecho) está vacía, se entiende que es por la necesidad de potencia de CPU que requerirían estos métodos, aunque no se descarta que en el futuro aparezca algún método en este cuadrante señalado.

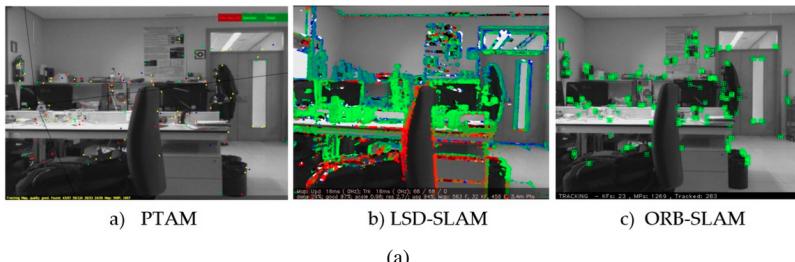


Figure 9: Diferencia entre mapas generados por distintos algoritmo de Visual SLAM (b)

4.4 MonoSLAM

El algoritmo de MonoSLAM (*Monocular SLAM*) [1] utiliza solamente una cámara RGB para la localización y mapeo de entornos desconocidos. Fue desarrollado en el año 2002 por Andrew Robinson. Para estimar la posición de la cámara utiliza un filtro extendido de Kalman (EKF) y la posición de una serie de puntos 3D. Este método requiere de una inicialización con al menos 4 puntos 3D conocidos que utilizará para calcular la posición de la cámara y la generación de nuevos puntos para el mapa.

El EKF, tiene un vector de estado compuesto de posición, orientación y velocidad de la cámara y además las coordenadas 3D de los puntos conocidos en un cierto momento, esto implica que el vector de estado irá aumentando de tamaño a medida que vayamos descubriendo

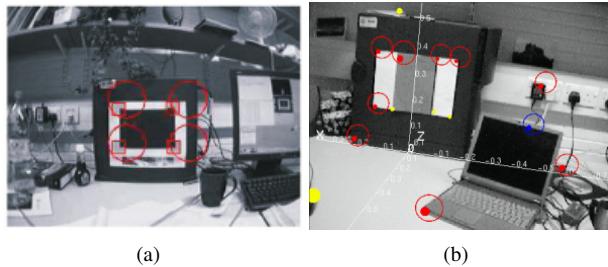


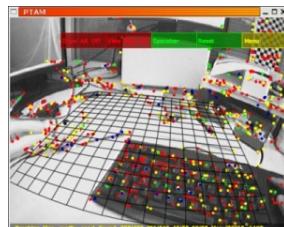
Figure 10: Inicialización de MonoSLAM con 4 puntos conocidos.

nuevos puntos 3D. El modelo de observación estará compuesto de las proyecciones de cada uno de los puntos 3D en el plano imagen. El uso de un EKF es apropiado, ya que se realizan iteraciones cada pocos milisegundos, y por tanto en intervalos de tiempo tan pequeños, el sistema puede aproximarse a un sistema lineal. Cuantas más iteraciones o frecuencia de muestreo la estimación mejora. En cada iteración se hace una detección de puntos de interés (esquinas con FAST) en la imagen actual de entrada, y obtendremos una serie de puntos que serán candidatos a ser el vector observación de los puntos que queremos seguir. Estos candidatos deberán ser filtrados, pues alguno puede ser un falsa esquina. Se utilizará una función de divergencia ZMSSD (*Zero Mean Sum of Squared Differences*) entre parches para determinar si el candidato es aceptable o no. Al utilizar sólo parches de unos pocos píxeles alrededor del candidato, estamos optimizando el computo ya que no requiere procesar toda la imagen. Aún así es posible que se acepten puntos candidatos que no sean apropiados. Para tratar de eliminar estos falsos positivos, [10] propuso una alternativa conocida como 1-Point RANSAC. MonoSLAM es recomendable para mapas con pocos puntos. Es muy sensible a movimientos bruscos y por tanto difícilmente podrá recuperarse de un secuestro. Si la hipótesis de partida no es correcta el filtro podría desestabilizarse y no llegar nunca a aproximar razonablemente el vector de estado.

4.5 PTAM

Parallel Tracking and Mapping. Es un algoritmo creado en 2007 por George Klein [11] que también calcula el *Tracking* y el *Mapping* como en MonoSLAM pero para ello utiliza 2 *Threads*, uno para calcular el posicionamiento de la cámara (*Tracking*) y el segundo para la generación del mapa (*Mapping*). Esta separación en dos hilos de ejecución se debe a que el *Tracking* necesita ser calculado en tiempo real para obtener una localización precisa, mientras que el *Mapping* puede demorarse más tiempo sin perjudicar a la localización de la cámara. Con las imágenes captadas en secuencia se van generando *Keyframes* o fotografías clave. Se genera un nuevo *Keyframe* a medida que la cámara se va desplazando. Los *Keyframes* se utilizan para la localización y para ir generando el mapa de puntos. Este algoritmo es recomendable para mapas con elevado número de puntos, es capaz de recuperarse fácilmente de un secuestro, extrae los puntos de interés mediante extracción de características como en MonoSLAM y trata de emparejarlos con los puntos extraídos de las imágenes anteriores. Como extractor de características utiliza el detector FAST. Se realizará una subdivisión de la imagen a distintas resoluciones, normalmente 4 niveles, lo que se conoce como pirámide de la imagen y se pasará un filtro FAST sobre esta pirámide para detectar los puntos

más característicos de la imagen. Cada *Keyframe* que se genera, contiene la imagen captada junto su pirámide y sus puntos de interés detectados. Cuando añadimos un *Keyframe*, se intenta localizar en este *Keyframe* los puntos que ya se encuentran en el mapa, en caso de no localizarlos se añaden nuevos puntos al mapa. Mientras no se añadan *Keyframes*, se intentará mejorar el mapa con los *Keyframes* disponibles optimizando con Bundle Adjustment. Se suele utilizar en entornos cerrados y pequeños y utiliza técnicas SFM. Muy utilizado también para realidad aumentada.



(a)

Figure 11: Nube de puntos característicos tomados con PTAM.

4.6 DTAM

Dense Tracking and Mapping. Es un método de reconstrucción de tipo denso que emplea el error fotométrico para poder trabajar en el dominio de la imagen [8]. La fase de localización (Tracking) se resuelve por una formulación alternativa a EKF utilizando ESM (Minimización Eficiente de segundo orden), de esta forma se puede ejecutar en paralelo. Para la reconstrucción del mapa emplea una metodología basada en la transformada de Radón. Cada píxel 3D será representado como un cubo, que se proyectará a cada una de las imágenes esclavas. Cuando la recta entre estos píxeles sea cero, el error fotométrico será nulo y entonces se podrá considerar que la proyección es correcta.

Para conseguir un buen rendimiento en tiempo real DTAM necesitará el uso GPU.



(a)

Figure 12: Ejemplo de mapa generado con DTAM. Todos los puntos forman parte del mapa.

4.7 SVO

FAST Semi-Direct Monocular Visual Odometry. Permite ser utilizado en ordenadores con poca potencia de computo debido a la rapidez del algoritmo. Es un método híbrido entre

los métodos de extracción de características y métodos directos [5]. Se asemeja a PTAM en que también utiliza dos *Threads* independientes, el primero para Tracking y el segundo para *Mapping*. En el proceso de Tracking, el algoritmo trata de minimizar el error fotométrico, pero para acelerar el proceso sólo tiene en cuenta ciertas partes de la imagen, unos parches de 4x4 alrededor de los píxeles que se han identificado como candidatos. Toma los puntos 3D visibles del fotograma anterior, los proyecta sobre la imagen actual, obtiene parches de dimensiones 4x4 alrededor de los píxeles y trata de hallar el mínimo error fotométrico de esos puntos que servirá para hallar el emparejamiento entre las características de dos frames. Calcula el desplazamiento entre imágenes de forma muy eficiente. Para la estimación del movimiento, trataremos de minimizar el error fotométrico de los parches de 4x4 anteriores. Como último paso haremos la minimización del error de reproyección clásica de los métodos basados en características para corregir los residuos que genere el paso anterior, los cuales podrían provocar la pérdida de ortogonalidad. En cuanto al *Mapping* utilizaremos un modelo gaussiano en torno al valor de profundidad real, cuando la incertidumbre de un parche decae, este es añadido al mapa. Un nuevo frame tiene posibilidad de convertirse en *Keyframe* si diverge lo suficiente del resto.

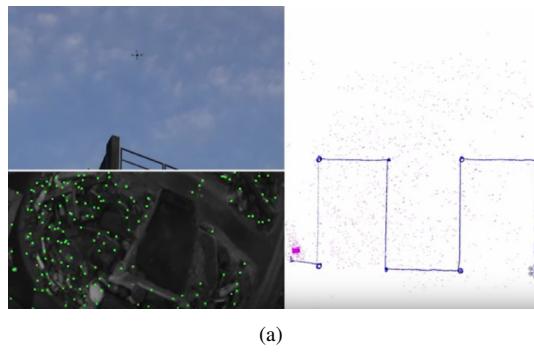
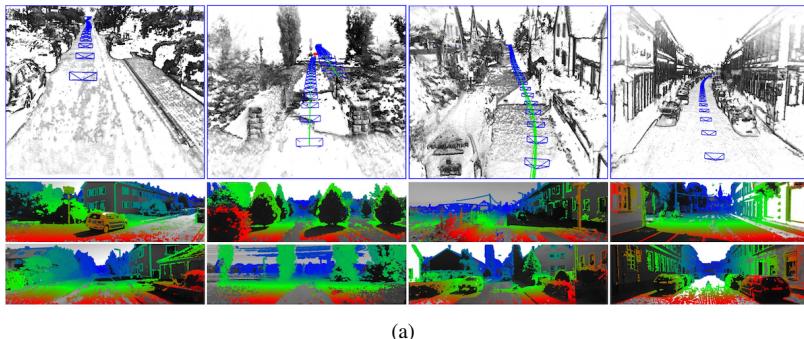


Figure 13: Mapa generado con un dron utilizando la técnica SVO.

4.8 LSD-SLAM

Large-Scale Direct Monocular SLAM La principal característica de este modelo es que trata de generar mapas del entorno a gran escala y consistentes. Utiliza para ello métodos directos. A demás de tener 2 hilos como PTAM uno para Tracking y otro para *Mapping*, existe un tercer componente encargado de estimar la profundidad del mapa.[6] El *Thread* de Tracking, parte de un *Keyframe* para calcular el desplazamiento, minimizando el error fotométrico que estará normalizado por la varianza. Utiliza una optimización ponderada de Gauss-Newton para medir la alineación entre frames. El *Thread* estimador de profundidad, inicializa el mapa de profundidad proyectando los puntos del *Keyframe* anterior. Las imágenes que no son *Keyframes* se usarán para refinar el *Keyframe* actual. Se añadirán nuevos píxeles al mapa de profundidad cuando se encuentren zonas de la imagen con suficiente separación estéreo. En cuanto al *Thread* dedicado al proceso de *Mapping*, cuenta con un mecanismo de cierre de bucle que se ejecutará cada vez que llegue un nuevo *Keyframe*. En cuanto a su inicialización, solo utiliza una única imagen para generar un mapa inicial de profundidad que irá convergiendo hacia unos valores de profundidad correctos a medida que la cámara

se vaya desplazando. Este método es capaz de funcionar en tiempo real en un PC, pero no funciona muy bien en dispositivos con limitada potencia de CPU.



(a)

Figure 14: Mapa generado con LSD-SLAM y cámara estéreo

4.9 ORB-SLAM

Es un algoritmo basado en extracción y emparejamiento de píxeles característicos mediante descriptores ORB, estos descriptores son más fiables que los parches tradicionales y por tanto permiten obtener mapas robustos y precisos tanto en escenarios de grandes dimensiones como en zonas pequeñas, sin embargo para su funcionamiento en tiempo real requiere la utilización de ordenadores con alta capacidad de proceso [8]. Puede ser utilizado con una cámara o dos e incluso con cámaras de profundidad RGBD. Para cierres de bucle y relocalización utiliza un modelo de bolsa de palabras [9]. Utiliza 3 *Threads*, el primero para Tracking, el segundo para *Mapping* y un tercero para detectar cierres de bucle. En el proceso de Tracking, se trata de calcular la posición actual a partir de los emparejamientos encontrados de los puntos 3D en el fotograma anterior, para ello utilizará los descriptores ORB. En caso de perdida, el robot podrá relocalizarse gracias a un modelo de bolsa de palabras que le permitirá encontrar *Keyframes* candidatos que concuerden con la observación actual (Figura 15(a)). En el proceso de *Mapping*, se inicializarán 2 mapas, uno por homografía y el segundo mediante una matriz fundamental. Los 2 mapas recibirán una puntuación y se elegirá como candidato para inicializar el mapa aquel que obtenga mayor puntuación. Cuando ya se dispone del mapa inicial, se procesan los *Keyframes* creando nuevos puntos 3D y se optimiza localmente el mapa mediante Bundle Adjustment. A su vez se genera un grafo donde cada *Keyframe* se corresponde con un vértice y un vértice estará unido a otro siempre y cuando los *Keyframe* tengan varios puntos 3D en común. Este grafo permite la eliminación de *Keyframe* redundantes (Figura 15(b)). En el proceso de Looping, se comprobará si se ha producido un cierre de bucle. Utilizando el grafo de *Keyframe* conectados y el modelo de bolsa de palabras se intenta encontrar *Keyframe* candidatos que tengan una apariencia similar a la imagen actual.

4.10 DSO

Direct Sparse Model. Está basado en optimizaciones continuas del error fotométrico sobre una ventana de frames recientes[10]. El inicio del Tracking, cuando se crea un nuevo

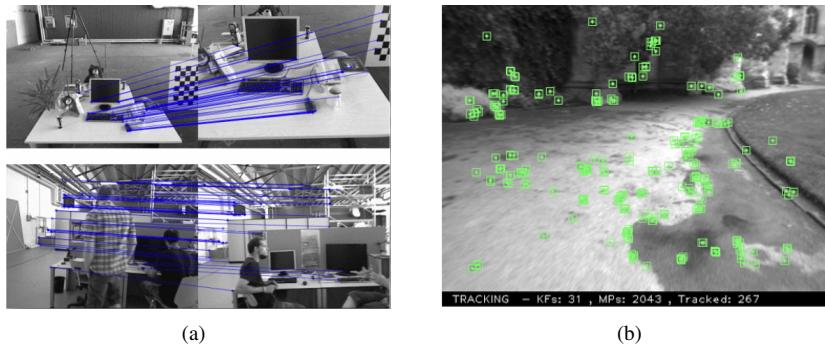


Figure 15: Localización de puntos característicos en 2 imágenes con ORB

Keyframe, todos los puntos activos son proyectados en el y ligeramente dilatados, creando así un mapa de profundidad semi denso. Nuevos frames son creados con respecto a este frame utilizando alineamiento directo de 2 frames, una pirámide multi escala y un modelo de movimiento constante a inicializar. Para la relocalización, se podrán trazar hasta 27 rotaciones pequeñas en diferentes direcciones. Esta recuperación de posición se consigue en el nivel más pequeño de la pirámide de la imagen. La creación de *Keyframes* es similar a ORB-SLAM, existen 3 criterios para determinar cuando se necesita un nuevo *Keyframe*.

1. Se creará un nuevo *Keyframe* (Figura 16(a)) cuando la imagen de entrada cambie notablemente con respecto al último *Keyframe*, esto se medirá con las diferencias de medias al cuadrado entre los píxeles.
 2. La traslación de la cámara causa occlusiones y des-occlusiones, lo cual indica que se deben generar nuevos *Keyframes*.
 3. Si el tiempo de exposición de la cámara cambia significativamente, se deberá tomar un nuevo *Keyframe*. Esto se mide por el factor de brillo relativo entre 2 frames.

En cuanto al rechazo de *Keyframes*, sigue la siguiente estrategia. Sean $I_1 \dots I_n$ un conjunto de *Keyframes* activos, siendo I_1 el más nuevo y I_n el más antiguo

1. Siempre se mantendrán los dos últimos *Keyframes* (I_1 e I_2)
 2. Frames con menos del 5% de sus puntos visibles en I_1 son descartados.
 3. Si mas de N frames están activos, se descartan (exceptuando I_1 e I_2) aquel que maximice un marcador de distancia $d(i,j)$ donde $d(i,j)$ es la distancia Euclídea entre *Keyframes* I_1 e I_j

Sobre el tratamiento de los puntos, siempre se tratará de mantener un numero fijo de puntos activos repartidos de forma uniforme entre el espacio y los frames activos. En un primer paso, se identifican N_p puntos candidatos en cada nuevo *Keyframe*. Los puntos candidatos no son inmediatamente sumados a la optimización, sino que son localizados individualmente en sucesivos frames generando una primera estimación del valor de profundidad que servirá como inicialización. En cuanto a la selección de puntos candidatos, se intentará seleccionar

aquellos puntos que están bien distribuidos en la imagen y tienen un valor elevado de gradiente con respecto a sus alrededores. Para obtener una distribución uniforme de puntos sobre la imagen, esta se divide en bloques de $d \times d$, de cada bloque se elegirá el píxel con el mayor gradiente siempre y cuando supere un umbral, de lo contrario no se selecciona el píxel de ese bloque. Los puntos candidatos son localizados en siguientes frames utilizando una búsqueda sobre la línea epipolar minimizando el error fotométrico. Una vez hallamos encontrado las coincidencias preparamos un valor de profundidad y la varianza asociada que se utilizará para restringir el intervalo de búsqueda en frames siguientes. Esta estrategia de localización está inspirada en LSD-SLAM. Por último, la activación de puntos candidatos, cuando un conjunto de puntos antiguos son marginados, nuevos puntos candidatos son activados para remplazarlos, siempre intentando mantener una distribución uniforme de puntos por toda la imagen¹¹.

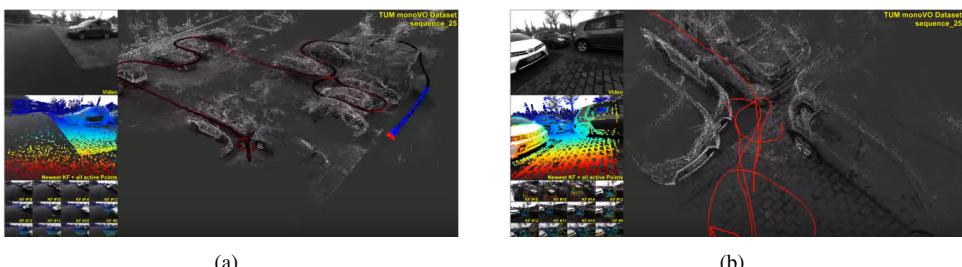


Figure 16: Mapa generado con DSO (a) Ligero error en la posición al volver al punto de partida (b).

4.11 RGB-D Visual SLAM

Las cámaras RGB-D, utilizadas en dispositivos como Kinect o smartphones en el Proyecto Tango, son capaces de proporcionar información 3D del entorno en tiempo real y por tanto estas cámaras también son utilizadas en Visual SLAM. A diferencia con los algoritmos del tipo monocular VisualSLAM, la escala del sistema de coordenadas es conocida para las cámaras RGB-D ya que son capaces de obtener las medidas y dimensiones de los objetos en 3D del entorno que le rodea. Para estimar el movimiento de la cámara se utiliza el algoritmo ICP *Iterative Closest Point*. La mayoría de cámaras que son capaces de medir la profundidad de los píxeles están creadas para entornos cerrados o de pequeñas dimensiones. Esto es debido a que estas cámaras proyectan un patrón de infrarrojos para medir la profundidad del entorno y es difícil detectar el patrón de infrarrojos emitidos en el exterior ya que la propia luz solar generaría ruido o perturbaciones en el rango infrarrojo. Además el rango de profundidad que pueden captar los sensores RGB-D está limitado de 7 a 9 metros. Para la localización, el movimiento relativo de la cámara es estimado identificando la localización de varios puntos característicos entre frames sucesivos. Utilizando estos puntos característicos se hace la estimación de los valores de una matriz de traslación. Con el algoritmo ICP y mapas de profundidad podremos optimizar esta matriz de traslación. También se utilizan métodos de localización basados en consistencia fotométrica similar a las técnicas utilizadas en el SLAM monocular.

¹¹<https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7898369>

en los métodos densos de Visual SLAM [10]. Para obtener un mapa geométricamente consistente, se utilizan varios algoritmos de optimización como pose-graph y deformation-graph. Pose-graph se utiliza para reducir el error acumulativo. Pose-graph es muy similar al bucle cerrado en los algoritmos de monocular VisualSLAM. En contraste con otros algoritmos, la estimación de mapa también está refinada. La optimización por Deformation Graph es muy utilizada para ciertos frames y la localización de la cámara es estimada con emparejamientos entre las imágenes RGB-D y el modelo reconstruido. Las APIs para RGB-D SLAM vienen incorporadas en dispositivos como Google Tango y Structure Sensor. Especialmente, Google Tango proporciona una estimación de resultado estable combinando también la información proporcionada por otros sensores internos del dispositivo.

5 Conclusiones

La robótica móvil es ya una realidad gracias a los algoritmos Visual SLAM que permiten estimar con mínimo error la localización y generación de mapas en entornos desconocidos. En este documento se han descrito algunos de estos algoritmos que ya están funcionando, pero se sigue investigando en la generación de nuevos métodos de navegación autónoma para conseguir mayor fiabilidad, robustez y exactitud de los cálculos. Dependiendo de las características del entorno o de los requisitos del problema que estemos tratando será más conveniente utilizar un algoritmo u otro. Por ejemplo si necesitásemos generar un mapa de gran exactitud, lo más conveniente sería utilizar DTAM, si por el contrario el mapa no fuese muy importante y la potencia del hardware fuese muy limitada podríamos utilizar SVO. Por ahora las limitaciones hardware hacen que en robótica móvil se opte por utilizar aquellas técnicas que requieren poca capacidad de cómputo (PTAM,SVO) ya que son fácilmente procesables por los microprocesadores de los actuales robots móviles. En el futuro y a medida que los robots tengan más capacidad de proceso, probablemente se impongan los métodos más robustos que realicen una localización más exacta y cuyos mapas sean muy fiables como podría ser el método ORB-SLAM o DSO. No obstante todavía queda un camino largo que avanzar en Visual SLAM, ya que algunos algoritmos no son del todo robustos en grandes espacios o entornos donde exista excesivo movimiento alrededor de la cámara, por ejemplo si nuestro robot se encontrase en un jardín frondoso, donde sopla una cierta brisa, le sería difícil al robot mapear el entorno ya que el movimiento de hojas y ramas podría generar inestabilidad en la estimación de la posición 3D de la cámara. Aunque la gran revolución se producirá cuando la mayoría de smartphones y cámaras estén equipadas con dispositivos que puedan medir la profundidad de las imágenes, como el proyecto Tango. Sin duda los cálculos de mapeo y posición se acelerarán y mejorará notablemente la exactitud de las estimaciones de posición. No sería de extrañar que próximamente apareciesen nuevos dispositivos periféricos que pudiesen ser controlados por el Smartphone, por ejemplo un nuevo tipo de aspiradora que no tuviese capacidad para realizar Visual SLAM por si sola, solo un par de motores que le permitan avanzar y girar. Si quisiésemos que esta aspiradora comenzase a aspirar de forma autónoma sólo tendríamos que colocar nuestro Smartphone en posición vertical sobre ella. El smartphone comenzaría a mapear la habitación y a dirigir la navegación de la aspiradora hasta que todo el suelo de la habitación quedase limpio. De esta forma todo el proceso de Visual SLAM de la aspiradora quedaría relegada al Smartphone. Y quien sabe, quizás el futuro de la conducción autónoma dependa de la capacidad con la que estén equipados para realizar Visual SLAM los cada vez más potentes Smartphones.

References

- [1] Javier Civera, Oscar G Grasa, Andrew J Davison, and JMM Montiel. 1-point ransac for extended kalman filtering: Application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 27(5):609–631, 2010.
- [2] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6), 2007.
- [3] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.
- [4] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *arXiv preprint arXiv:1607.02565*, 2016.
- [5] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2017.
- [6] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.
- [7] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.
- [8] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.
- [9] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.
- [10] Sei Ikeda Takafumi Taketomi, Hideaki Uchiyama. Visual slam algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, Junio 2017.